Fitbit Project

Gaayathri Iyer

9/20/2019

Fitness Tracker Project

Synopsis

The purpose of the project is to find how well the Weight Lifting Exercise is done. The data set includes the data from accelerometers on the belt, forearm, arm, and dumbell of six participants. They were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). All classes except Class A correspond to mistakes.

Load the data for analysis

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(url, destfile = "FitBitTraining.csv")
df <- read.csv("FitBitTraining.csv")
cat("Raw Data Dimension - ", dim(df), "\n\n")
```

```
## Raw Data Dimension -  19622 160
```

```
## First few column names with NAs
```

```
## [1] "max_roll_belt"        "max_picth_belt"       "min_roll_belt"
## [4] "min_pitch_belt"       "amplitude_roll_belt"  "amplitude_pitch_belt"
```

```
length(colNa)
```

```
## [1] 67
```

```
## There are  67  columns with NAs.
```

```
##  Let us see the total NAs in few columns to decide if these columns are important.
```

```
##        max_roll_belt       max_picth_belt        min_roll_belt
##                19216                19216                19216
##       min_pitch_belt  amplitude_roll_belt amplitude_pitch_belt
##                19216                19216                19216
```

We can see that the NAs in the columns are 98%. This shows that we can ignore these columns for our model prdiction.

Clean the data

Remove all Near Zero Value(NZV) columns from the data set as they are not good predictors.

```
df[is.na(df)] <- 0
df <- df[,-nearZeroVar(df)]
df <- df[,-c(1:6)]
cat("Cleaned Data Dimension - ", dim(df))
```

```
## Cleaned Data Dimension -  19622 53
```

Prepare training set for model training

Let us partition the training set for model training ie., split training set into training and testing sets to train the model

```
inTrain <- createDataPartition(df$classe, p=0.7, list=FALSE)
training <- df[inTrain,]
testing <- df[-inTrain,]
cat("Model Training : Data Dimension - ", dim(training), "\n")
```

```
## Model Training : Data Dimension -  13737 53
```

```
cat("Model Testing  : Data Dimension - ", dim(testing))
```

```
## Model Testing  : Data Dimension -  5885 53
```

Now that the training data is split into 2 sets for model training and testing, we need to choose the best model to train our data set. Let us try the classification models Parallel Random Forest and Gradient Boosting with cross-validation to pick the best one for our scenario.

Parallel Random Forest Model

```
clusters <- makeCluster(4)
registerDoParallel(clusters)
fitRF <- train(classe ~ ., data=training, method="parRF", trControl=trainControl("cv", number=4), metri
```

Summary of Random Forest method used

```
fitRF$finalModel
```

```
##
## Call:
##  randomForest(x = "x", y = "y", ntree = 125, mtry = 2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
```

```
fitRF
```

```
## Parallel Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 10303, 10303, 10303, 10302
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9912644  0.9889488
##   27    0.9907548  0.9883048
##   52    0.9804909  0.9753216
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Test the model

```
predictFit <- predict(fitRF, newdata=testing)
cmRF <- confusionMatrix(predictFit, testing$classe)
cmRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    7    0    0    0
##          B    3 1125   11    0    0
##          C    0    7 1014   26    0
##          D    0    0    1  938    3
##          E    0    0    0    0 1079
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9901
##                  95% CI : (0.9873, 0.9925)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9875
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9877   0.9883   0.9730   0.9972
## Specificity            0.9983   0.9971   0.9932   0.9992   1.0000
## Pos Pred Value         0.9958   0.9877   0.9685   0.9958   1.0000
## Neg Pred Value         0.9993   0.9971   0.9975   0.9947   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1912   0.1723   0.1594   0.1833
## Detection Prevalence   0.2851   0.1935   0.1779   0.1601   0.1833
## Balanced Accuracy      0.9983   0.9924   0.9908   0.9861   0.9986
```

From the above result we see that

```
## Accuracy of the model is  99.01444 %
```

```
## Out of sample error is  0.9855565 %
```

Gradient Boosting Model

```
fitGBM <- train(classe ~ ., data=training, method="gbm", trControl=trainControl("cv", number=4), verbos
```

Summary of Gradient Boosting method used

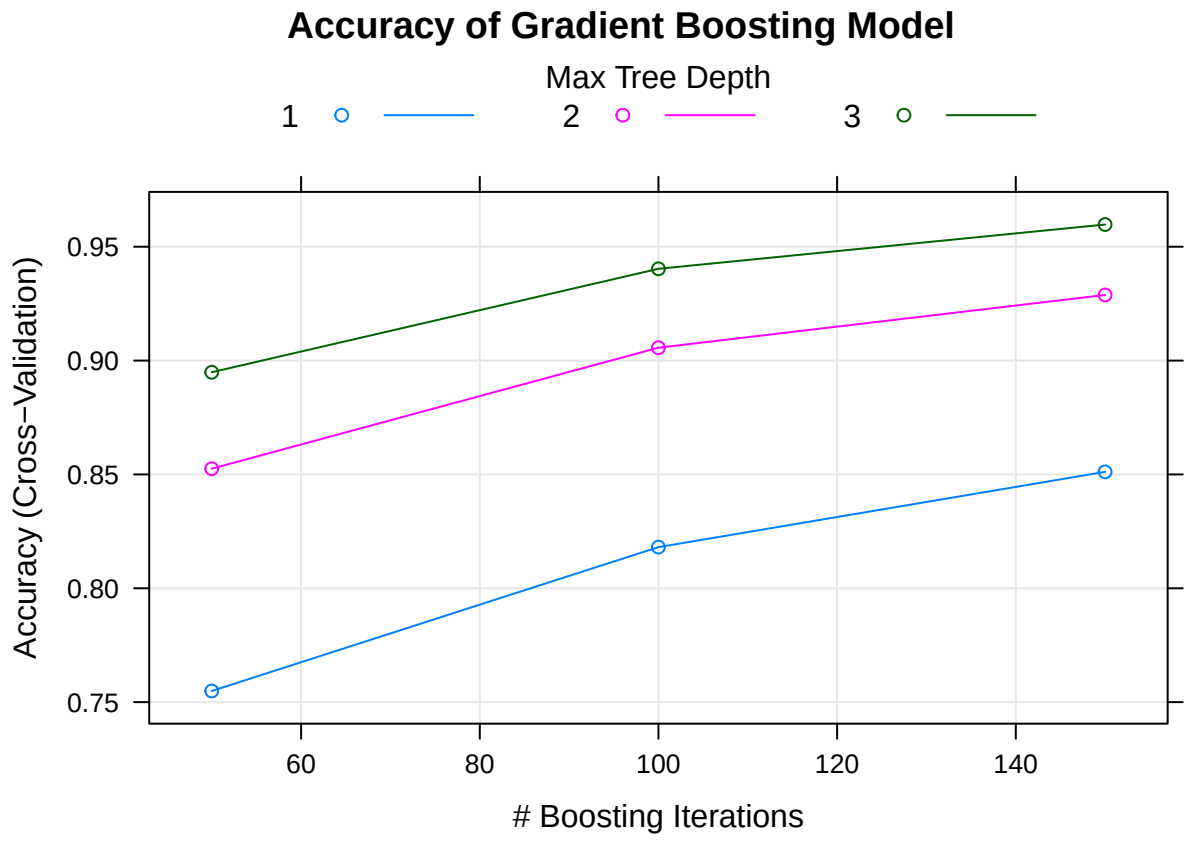
```
fitGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 51 had non-zero influence.
```

```
(fitGBM)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 10303, 10303, 10302, 10303
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7548950  0.6892440
##   1                  100      0.8180821  0.7696867
##   1                  150      0.8511317  0.8115744
##   2                   50      0.8525151  0.8131284
##   2                  100      0.9056562  0.8806119
##   2                  150      0.9288052  0.9099120
##   3                   50      0.8948823  0.8669106
##   3                  100      0.9403069  0.9244612
##   3                  150      0.9597436  0.9490688
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```



Test the model

```
predictFit <- predict(fitGBM, newdata=testing)
cmGBM <- confusionMatrix(predictFit, testing$classe)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   27    0    2    3
##          B   22 1082   51    3    9
##          C    3   28  961   44    9
##          D    1    1   10  911   12
##          E    1    1    4    4 1049
##
## Overall Statistics
##
##                Accuracy : 0.9601
##                  95% CI : (0.9547, 0.9649)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9495
##
##  Mcnemar's Test P-Value : 1.281e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9500   0.9366   0.9450   0.9695
## Specificity            0.9924   0.9821   0.9827   0.9951   0.9979
## Pos Pred Value         0.9809   0.9272   0.9196   0.9743   0.9906
## Neg Pred Value         0.9936   0.9879   0.9866   0.9893   0.9932
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1839   0.1633   0.1548   0.1782
## Detection Prevalence   0.2853   0.1983   0.1776   0.1589   0.1799
## Balanced Accuracy      0.9881   0.9660   0.9597   0.9701   0.9837
```

```
## Accuracy of GBM is  96.0068 %
```

Conclusion

```
## From the above two models' accuracies we can see that the Random Forest model has a better predictio
```

Variable Importance - The top predictors of the model

```
varImp(fitRF)
```

```
## parRF variable importance
##
##   only 20 most important variables shown (out of 52)
##
```

```
##                   Overall
## roll_belt          100.00
## yaw_belt            85.03
## magnet_dumbbell_z   72.55
## magnet_dumbbell_y   68.70
## pitch_forearm       67.10
## pitch_belt          59.17
## magnet_dumbbell_x   56.08
## roll_forearm        55.17
## accel_dumbbell_y    49.41
## roll_dumbbell       45.28
## accel_belt_z        45.10
## magnet_belt_z       44.63
## magnet_belt_y       37.45
## roll_arm            36.15
## accel_dumbbell_z    35.56
## accel_forearm_x     33.14
## gyros_dumbbell_y    32.97
## magnet_arm_y        30.34
## magnet_forearm_z    29.64
## gyros_belt_z        29.23
```

Apply the trained model to the Test Data

```
url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url, destfile = "FitBitTest.csv")
dfTest <- read.csv("FitBitTest.csv")
cat("Raw Test Data Dimension - ", dim(dfTest))
```

```
## Raw Test Data Dimension -  20 160
```

```
dfTest[is.na(dfTest)] <- 0
dfTest <- dfTest[,-nearZeroVar(dfTest)]
dfTest <- dfTest[,-c(1:6)]
cat("Cleaned Test Data Dimension - ", dim(dfTest))
```

```
## Cleaned Test Data Dimension -  20 53
```

Now predict the Test Data variable ('classe') using our trained model

```
prdTest <- predict(fitRF, newdata = dfTest)
prdTest
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

```
stopCluster(clusters)
```